Azure Monitoring Hackathon Proctor Guide

A screenshot of a cell phone

Description automatically generated

Version 1.0

Author: Rob Kuehfus

Please send feedback to here - <https://github.com/rkuehfus/AzureMonitoringHackathon>

Contents

[Getting Started 2](#_Toc530988600)

[Deployment Steps 4](#_Toc530988601)

[Troubleshooting 7](#_Toc530988602)

[Validation 8](#_Toc530988603)

# Hack Setup

* Deploy Infra using PowerShell \ CLI and an ARM Template
  + Setup Azure Module on Windows PowerShell  
    <https://docs.microsoft.com/en-us/powershell/azure/install-azurerm-ps?view=azurermps-6.9.0>
  + Setup Azure CLI  
    <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest>
  + Install Visual Studio Code and Extensions (depending on your tool of choice)
    - Azure Resource Manager Tools - <https://marketplace.visualstudio.com/items?itemName=msazurermtools.azurerm-vscode-tools>
    - ARM snippets - <https://marketplace.visualstudio.com/items?itemName=artofshell.armsnippet>
    - PowerShell - <https://marketplace.visualstudio.com/items?itemName=ms-vscode.PowerShell>
    - Azure CLI Tools –   
      <https://marketplace.visualstudio.com/items?itemName=ms-vscode.azurecli>
  + Download Azure Monitoring Hackathon Setup Guide and follow the instructions  
    <https://github.com/rkuehfus/AzureMonitoringHackathon/raw/master/Student/Guides/Azure%20Monitoring%20Hackathon%20Deployment%20Guide.docx>

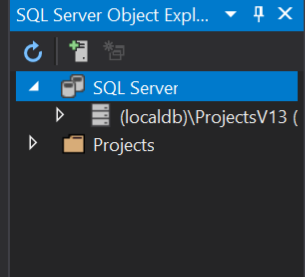
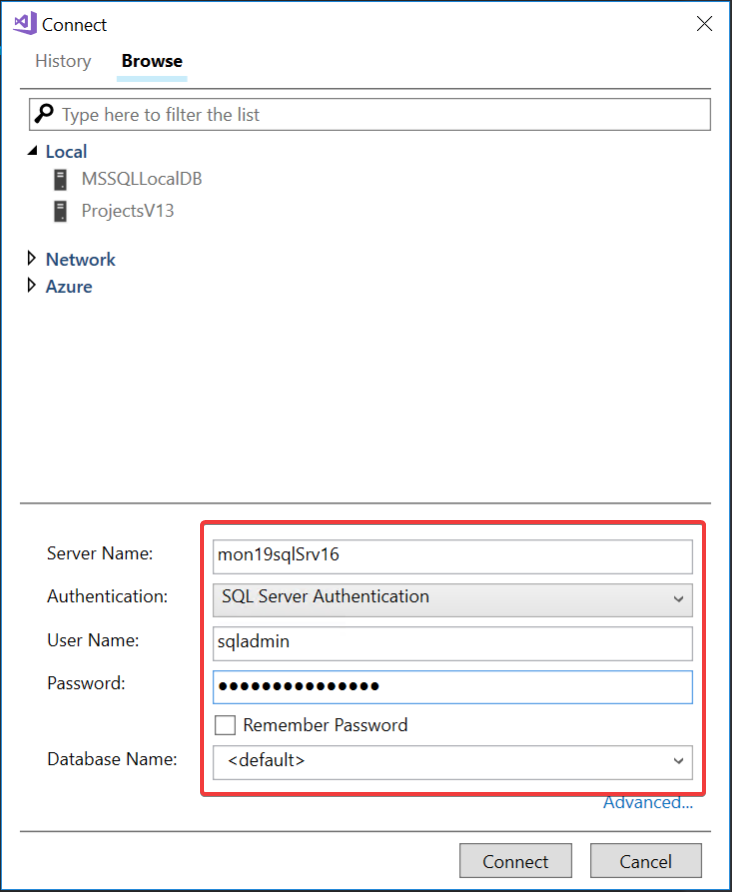
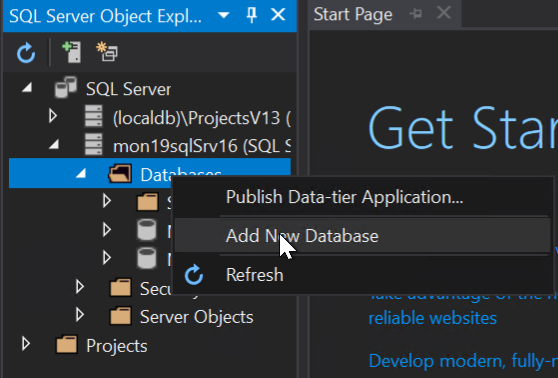
Navigate to <https://github.com/rkuehfus/AzureMonitoringHackathon> and download the repository as a zip file to your local disk

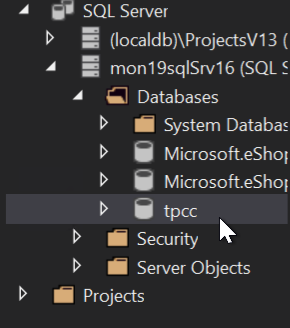
## Guidance

Biggest issues I have seen when deploying the hack are around compute quotas, not having contributor role membership to the subscription and local machine issues. If you run into a student who cannot seem to get their computer working with either the Azure CLI or PowerShell have them build a VM in Azure and configure this VM to do the deployment. Another option would be to have them run the deployment from the Azure Cloud Shell. Remember to copy up the ARM template and parameters JSON files before kicking off the deployment.

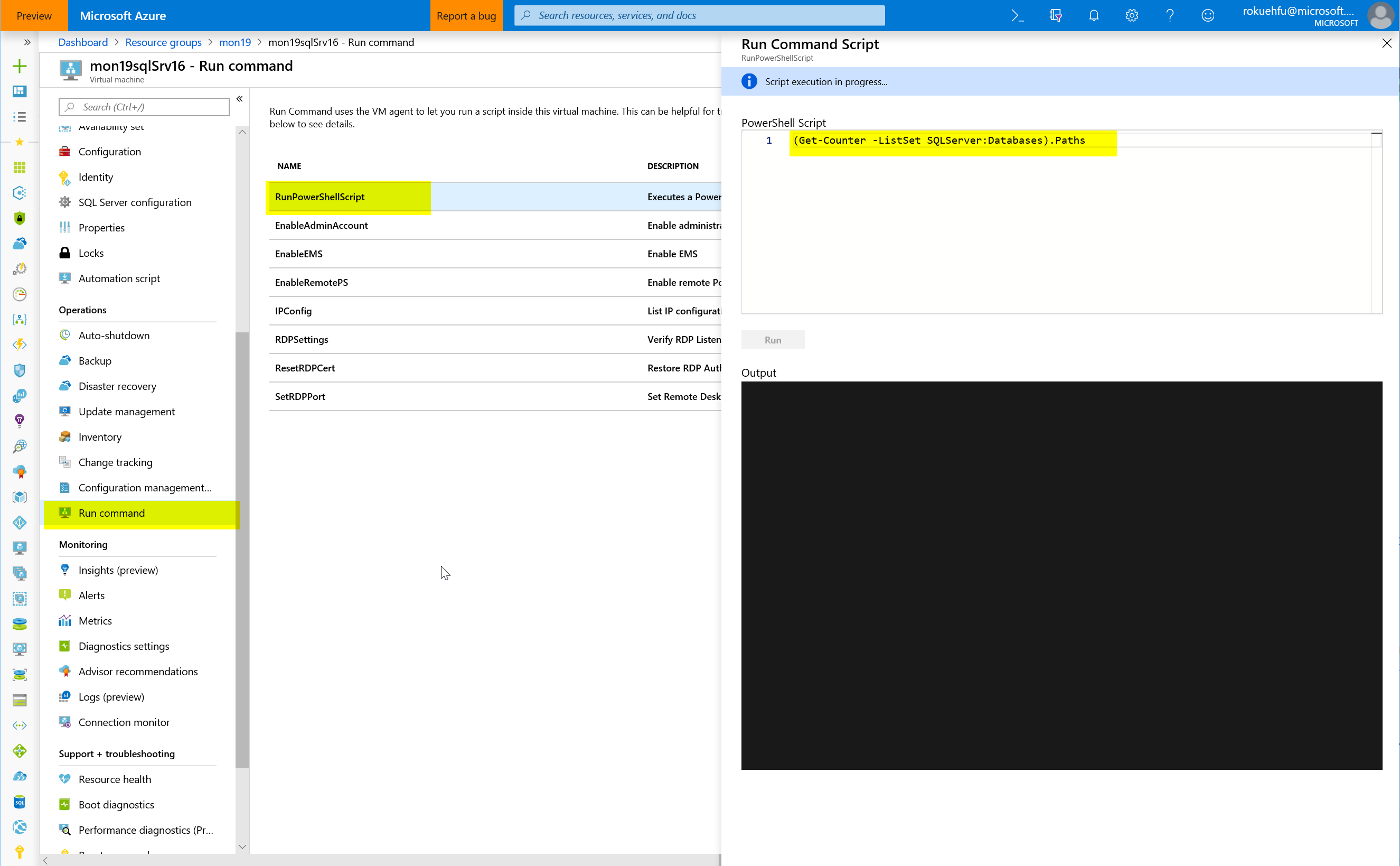
I highly recommend your students make sure they have an Azure Subscription they have contributor role access to before day one of the hack.

# Day 1: Monitoring and Alert Rule Challenge

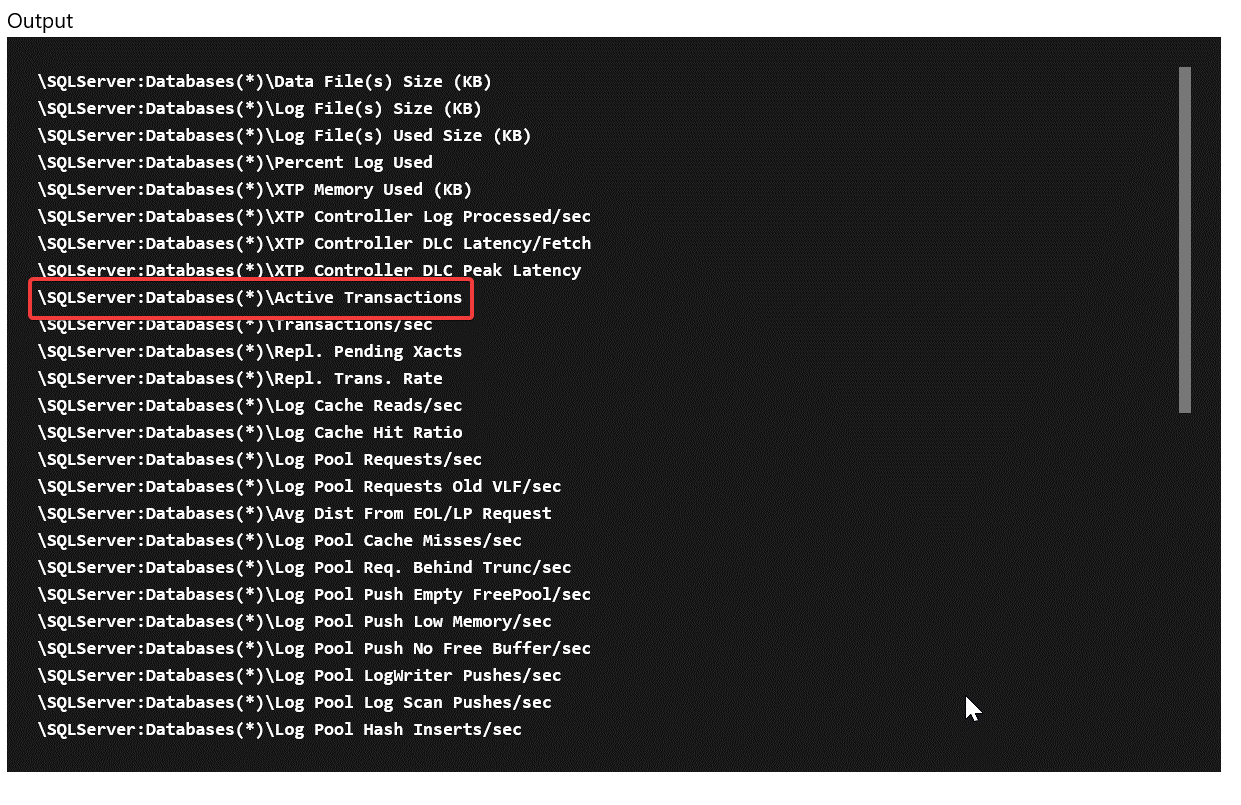
* Create an empty database called “tpcc” on the SQL Server  
  Note: Use SQL Auth with the username being sqladmin and password being whatever you used during deployment
  + Connect (RDP) to the Visual Studio Server (xxxxxVSSrv17) using its public IP address and open Visual Studio. Create an account if you don’t have one.
  + VS has view called SQL Server Object Explore that can be used to create and delete SQL databases on the SQL server  
    
  + Connect to the database server, make sure to use sqladmin and the password you stored in the key vault during deployment  
      
      
    Once connected create a new database called “tpcc”



* From the ARM template, send the below guest OS metric to Azure Monitor for the SQL Server
  + Add a Performance Counter Metric for
    - Object: SQLServer:Databases
    - Counter: Active Transactions
    - Instance:tpcc
  + Hint: <https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/metrics-store-custom-guestos-resource-manager-vm>
  + First, figure out the correct format for the counter use the run command on the SQL Server in the Azure portal and run –



Run the command - (Get-Counter -ListSet SQLServer:Databases).Paths

Once its finished, review the results (scroll up) and copy the output for the SQLServer:Databases counter.  


\SQLServer:Databases(\*)\Active Transactions

Then change it to target just your specific database

\SQLServer:Databases(tpcc)\Active Transactions

**Tip:** Share the following link to help lead them to how to find the counter

<https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.diagnostics/get-counter?view=powershell-5.1>

Next, once you have the counter you need to modify the ARM template for the SQL Server to add the collection of this counter that sends it to Azure Monitor using the Azure monitor data sink

Add this JSON to the SQL Server code

Verify the identity is present in the template (add it if its missing)

"identity": {

"type": "SystemAssigned"

},

And the missing counter below

"resources":[

{

"type": "extensions",

"name": "Microsoft.Insights.VMDiagnosticsSettings",

"apiVersion": "2015-05-01-preview",

"location": "[resourceGroup().location]",

"dependsOn": [

"[concat('Microsoft.Compute/virtualMachines/', concat(parameters('envPrefixName'), 'sqlSrv16'))]"

],

"properties": {

"publisher": "Microsoft.Azure.Diagnostics",

"type": "IaaSDiagnostics",

"typeHandlerVersion": "1.4",

"autoUpgradeMinorVersion": true,

"settings": {

"WadCfg": {

"DiagnosticMonitorConfiguration": {

"overallQuotaInMB": 4096,

"DiagnosticInfrastructureLogs": {

"scheduledTransferLogLevelFilter": "Error"

},

"Directories": {

"scheduledTransferPeriod": "PT1M",

"IISLogs": {

"containerName": "wad-iis-logfiles"

},

"FailedRequestLogs": {

"containerName": "wad-failedrequestlogs"

}

},

"PerformanceCounters": {

"scheduledTransferPeriod": "PT1M",

"sinks": "AzMonSink",

"PerformanceCounterConfiguration": [

{

"counterSpecifier": "\\Memory\\Available Bytes",

"sampleRate": "PT15S"

},

{

"counterSpecifier": "\\Memory\\% Committed Bytes In Use",

"sampleRate": "PT15S"

},

{

"counterSpecifier": "\\Memory\\Committed Bytes",

"sampleRate": "PT15S"

},

{

"counterSpecifier": "\\SQLServer:Databases(tpcc)\\Active Transactions",

"sampleRate": "PT15S"

}

]

},

"WindowsEventLog": {

"scheduledTransferPeriod": "PT1M",

"DataSource": [

{

"name": "Application!\*"

}

]

},

"Logs": {

"scheduledTransferPeriod": "PT1M",

"scheduledTransferLogLevelFilter": "Error"

}

},

"SinksConfig": {

"Sink": [

{

"name": "AzMonSink",

"AzureMonitor": {}

}

]

}

},

"StorageAccount": "[variables('storageAccountName')]"

},

"protectedSettings": {

"storageAccountName": "[variables('storageAccountName')]",

"storageAccountKey": "[listKeys(variables('accountid'),'2015-06-15').key1]",

"storageAccountEndPoint": "https://core.windows.net/"

}

}

},

{

"type": "extensions",

"name": "WADExtensionSetup",

"apiVersion": "2015-05-01-preview",

"location": "[resourceGroup().location]",

"dependsOn": [

"[concat('Microsoft.Compute/virtualMachines/', concat(parameters('envPrefixName'), 'sqlSrv16'))]"

],

"properties": {

"publisher": "Microsoft.ManagedIdentity",

"type": "ManagedIdentityExtensionForWindows",

"typeHandlerVersion": "1.0",

"autoUpgradeMinorVersion": true,

"settings": {

"port": 50342

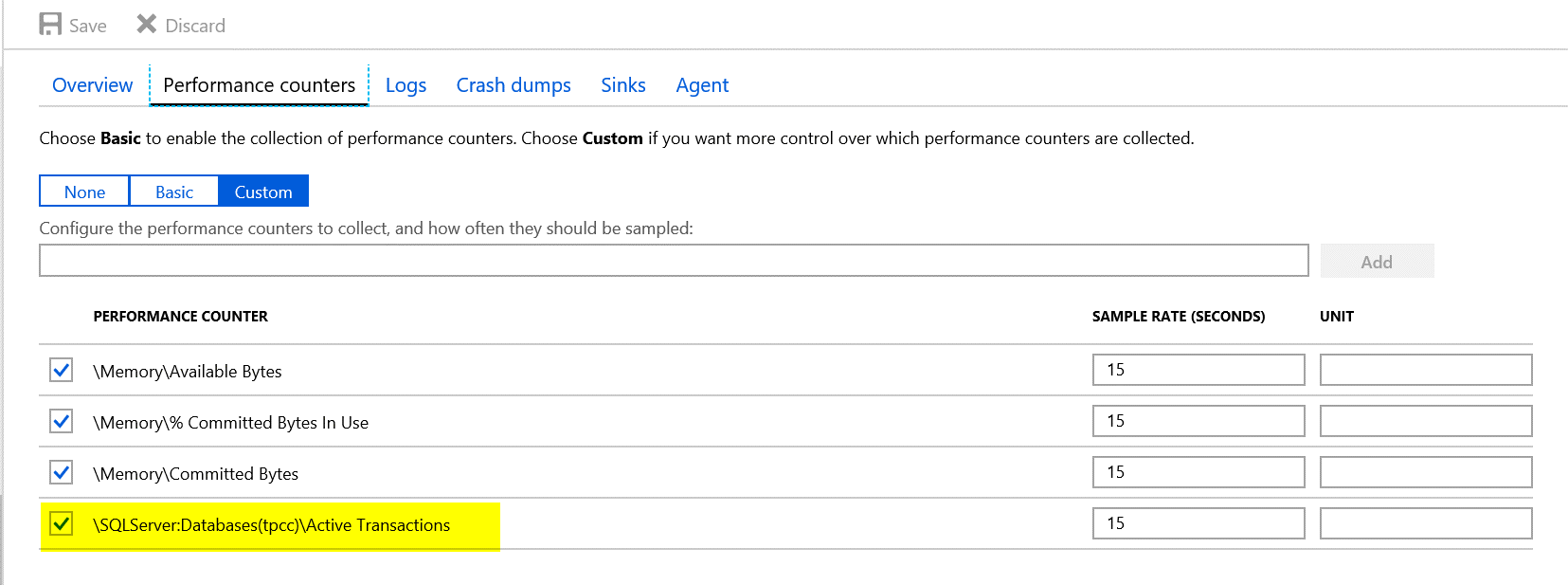
}

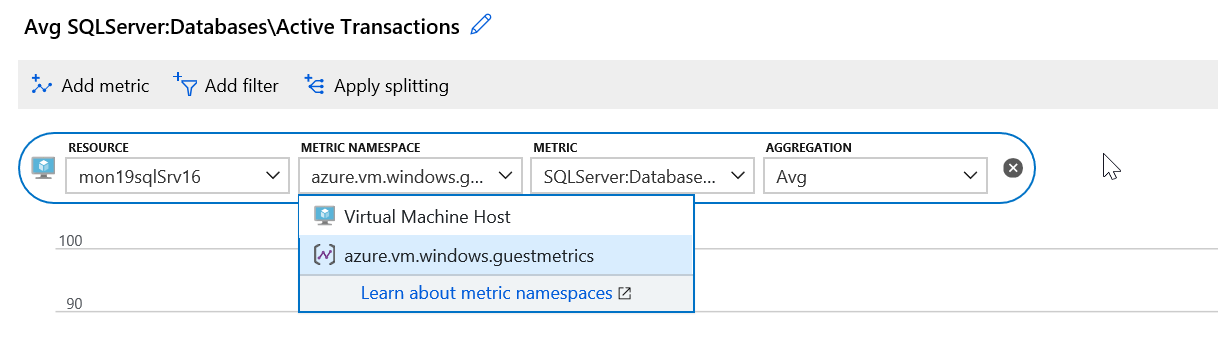
}

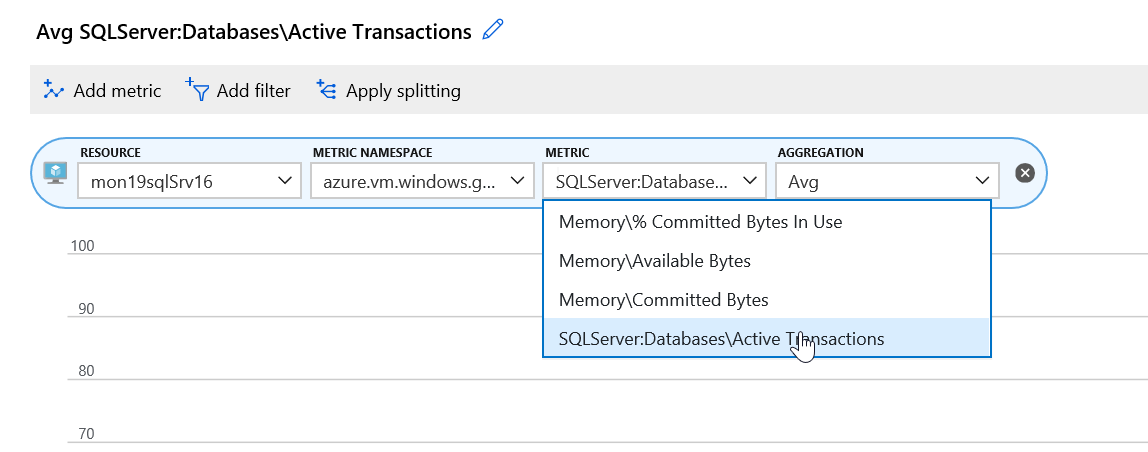
}

]

Save and redeploy.

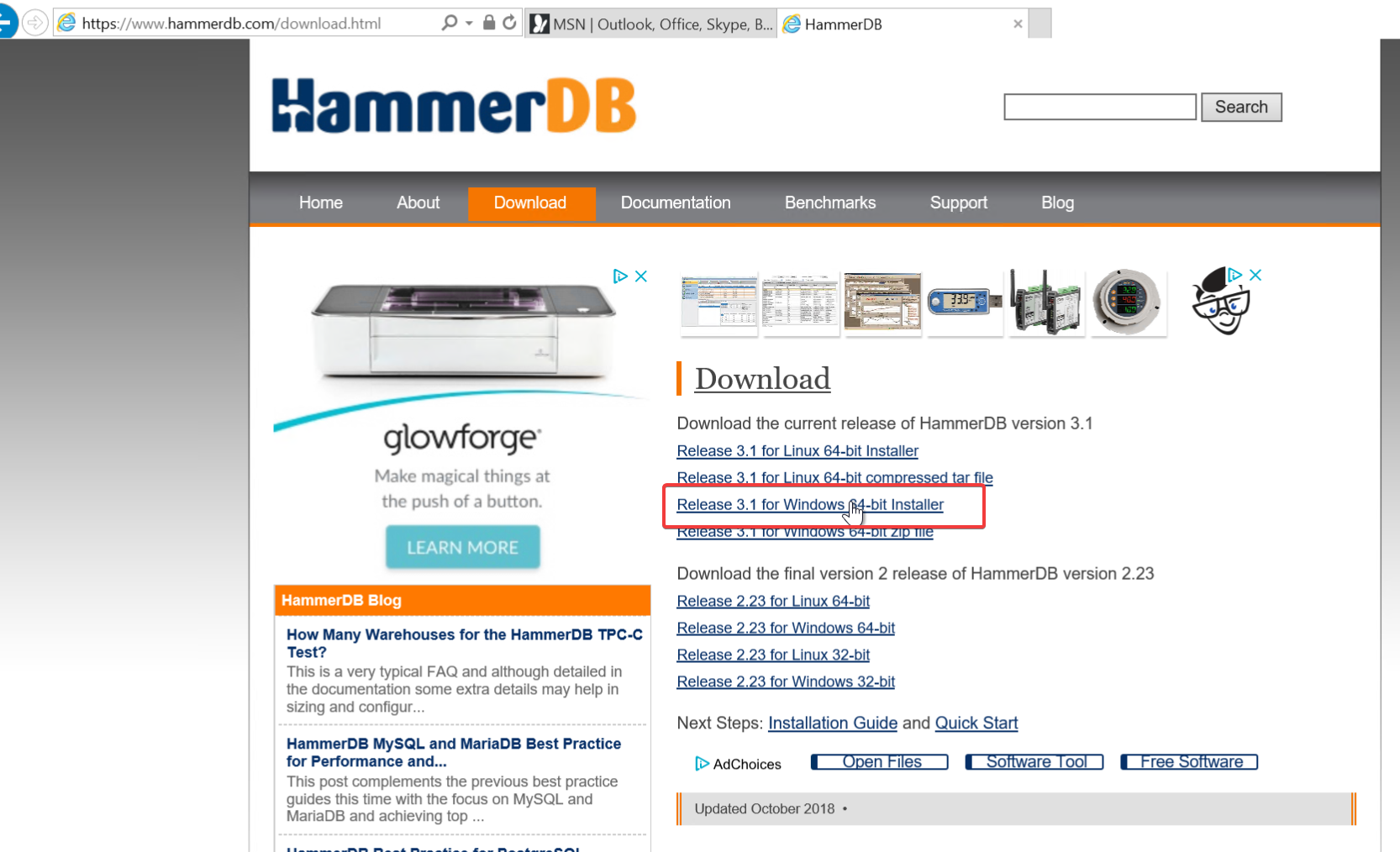
Once redeployed, go to metrics and check to make sure you are seeing the new metrics.   


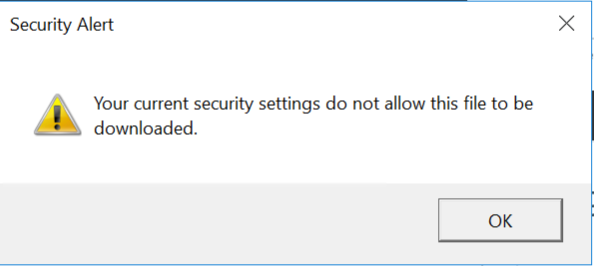




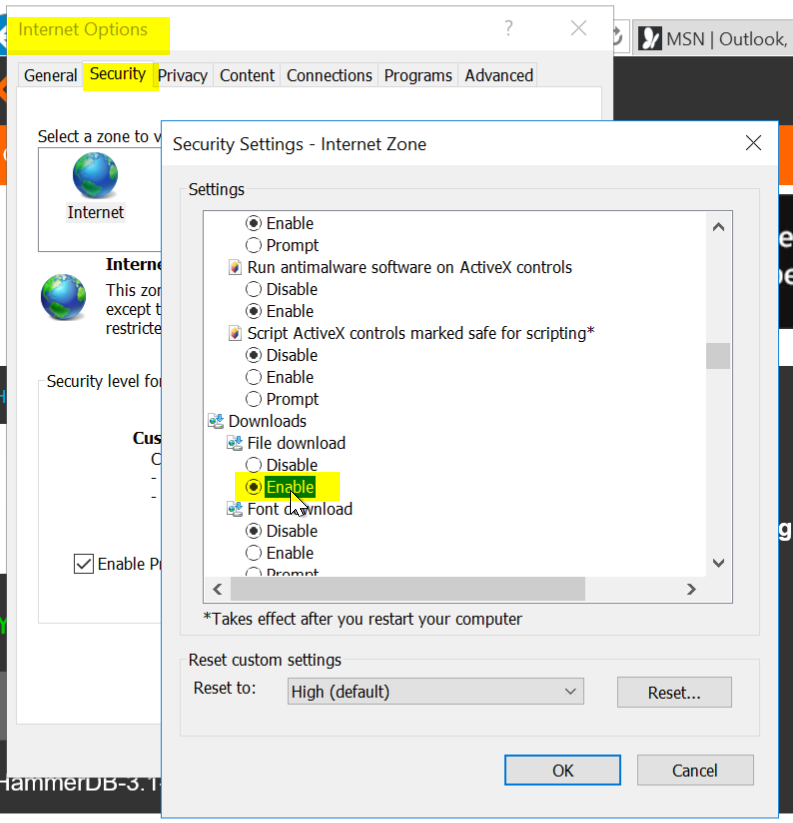
**Tip:** A bunch of OS metrics are configured already under the scale set as a sample.

* Download and Install HammerDB tool on the Visual Studio VM. Note: I copy of these instructions are in the student folder under “AzureMonitoringHackathon\Student\Guides\Day-1”
  + [www.hammerdb.com](http://www.hammerdb.com/)

From the Visual Studio Server, download the latest version of HammerDB  




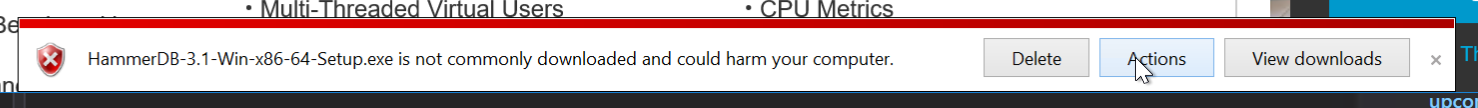
If you get this Security Warning, go to Internet Options.



Security \ Security Settings \ Downloads \ File download \ Enable

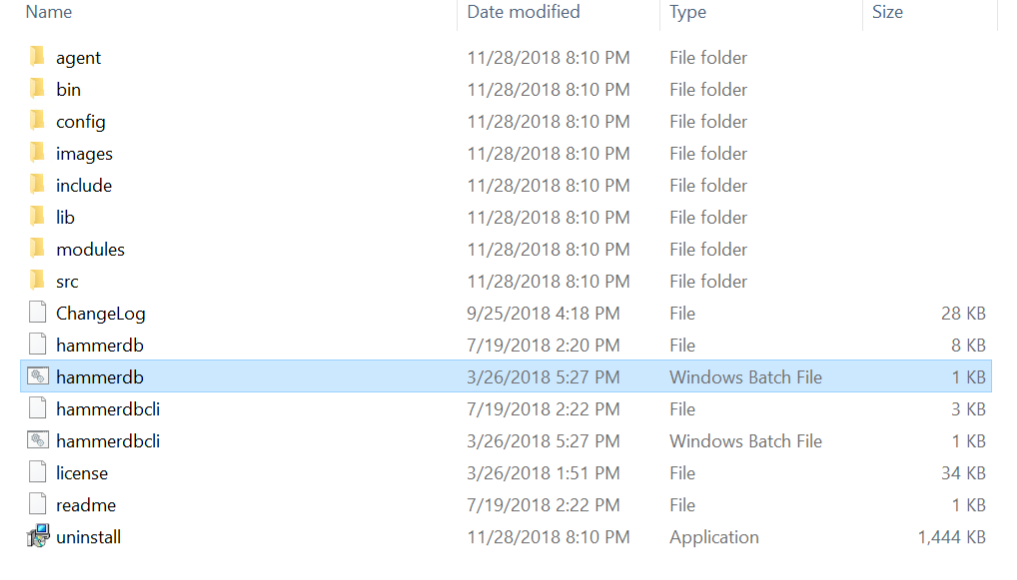
Click OK

Try again

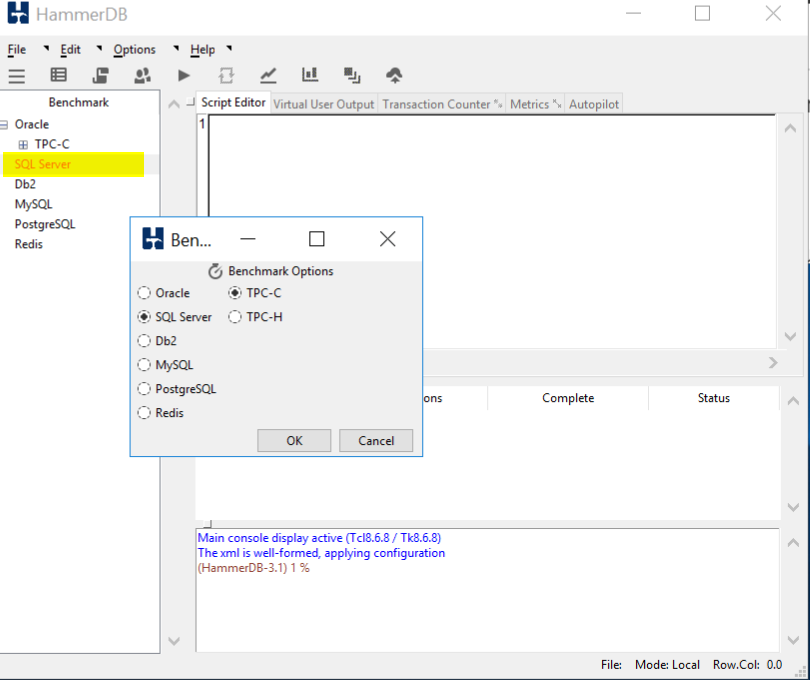


Click Actions and accept the warnings

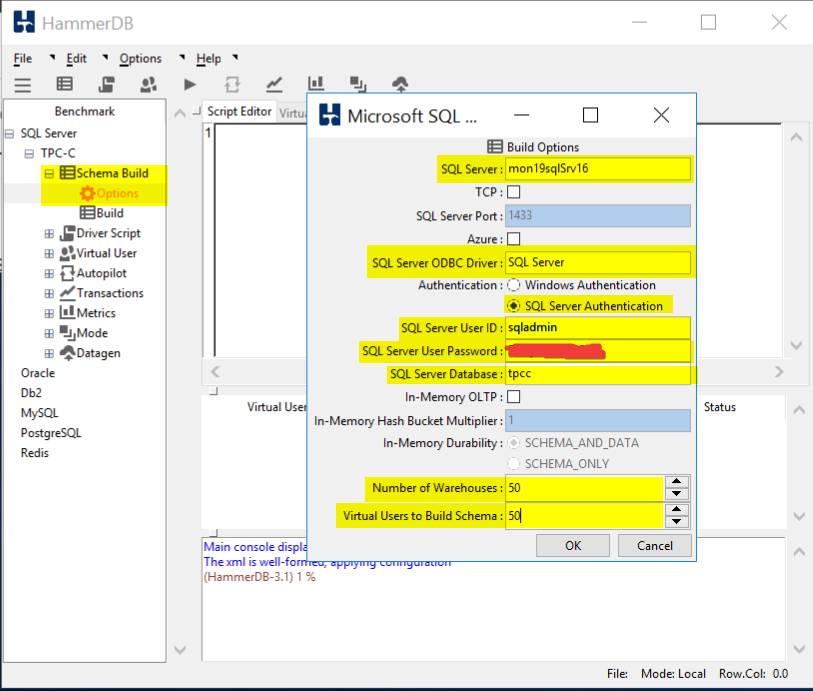
Tip: If you end up closing HammerDB you have to go to C:\Program Files\HammerDB-3.1 and run the batch file



* Use HammerDB to create transaction load



Double click on SQL Server and click OK, and OK on the confirm popup



Drill into SQL Server \ TPC-C \ Schema Build and double click on Options

Modify the Build Options for the following:

SQL Server: Name of your SQL Server

SQL Server ODBC Driver: SQL Server

Authentication: SQL Server Authentication

SQL Server User ID: sqladmin

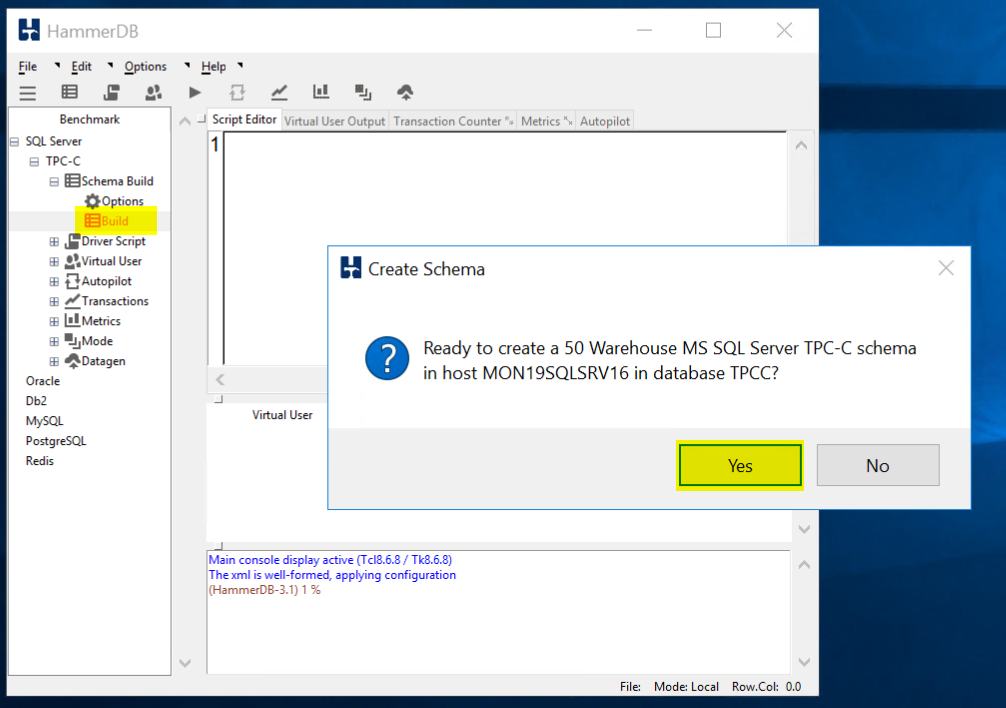
SQL Server User Password: <password you stored in the Key Vault>

SQL Server Database: tpcc

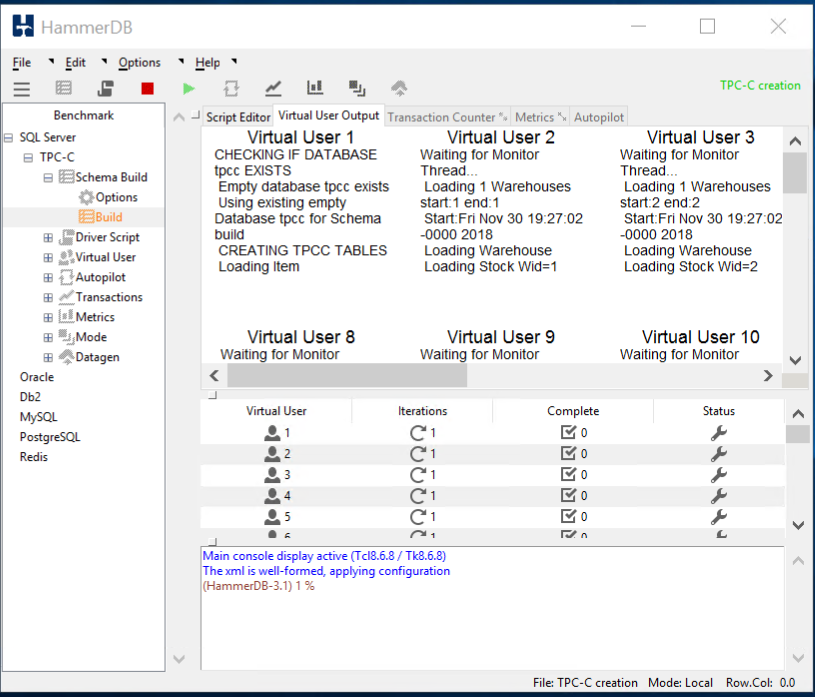
Number of Warehouses: 50

Virtual Users to Build Schema: 50

Note: Setting the last two at 50 should generate enough load to trip a threshold and run long enough for you to graph

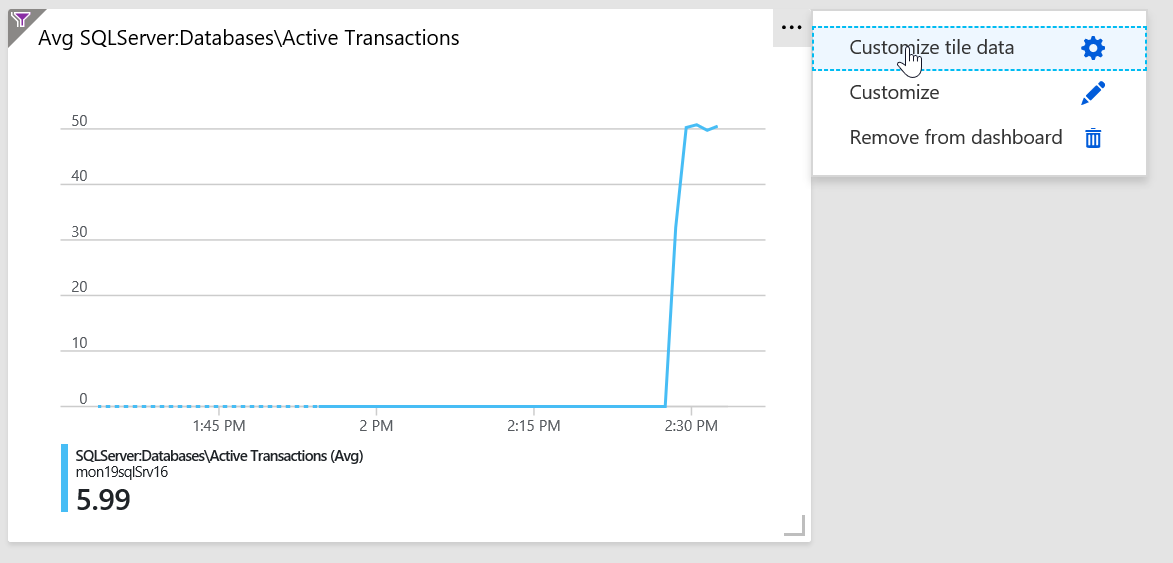


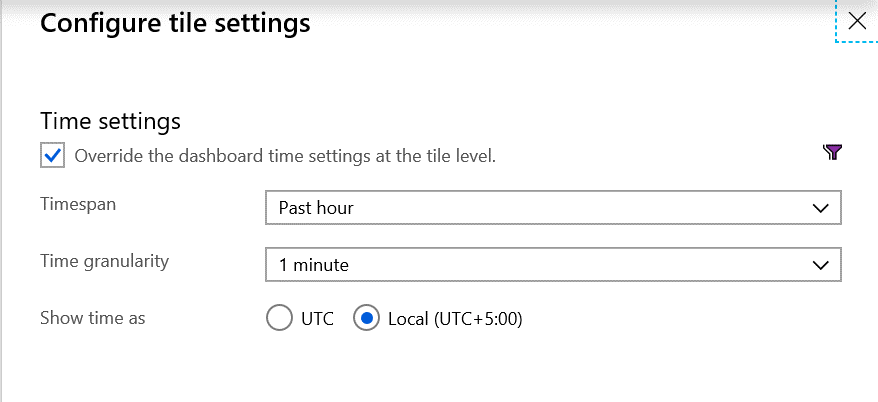
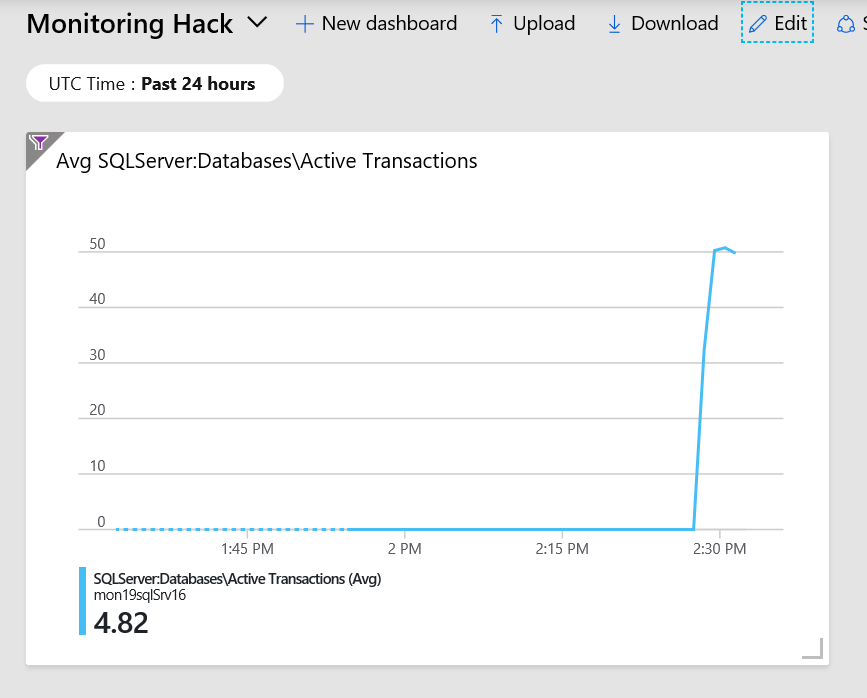
Double click on Build and Click Yes to kick of a load test.



When the test is running it should like this.

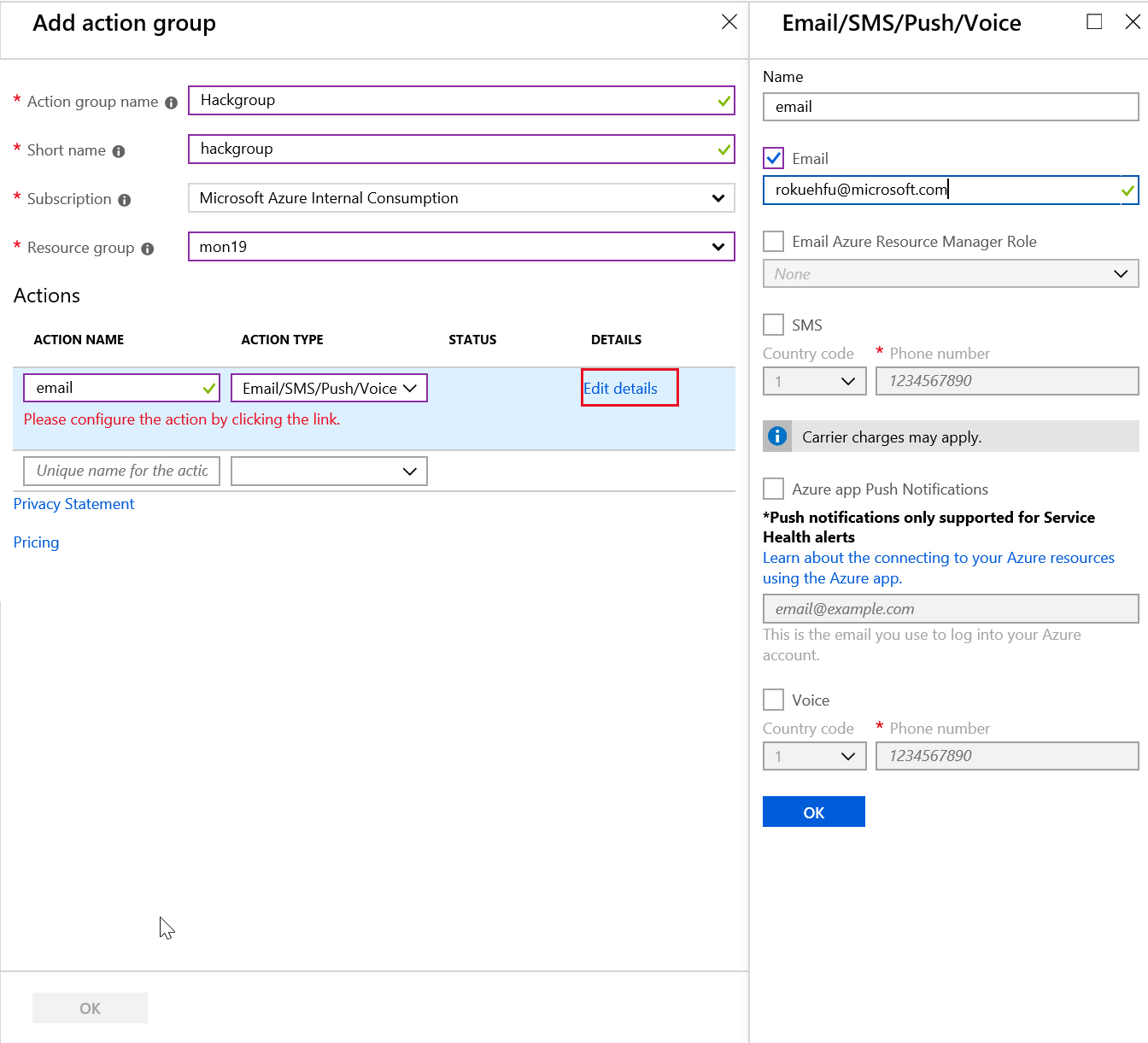
TIP: If you would like to run a second test you **must** first delete the database you created and recreate it. HammerDB will not run a test against a database that has data in it. When you run a test is fills the database with a bunch of sample data.

* From Azure Monitor,
  + Create a graph for the SQL Server Active Transactions and Percent CPU and pin to your Azure Dashboard
  + Note: I had to customize the dashboard once I pinned it to a new Azure Dashboard  
    

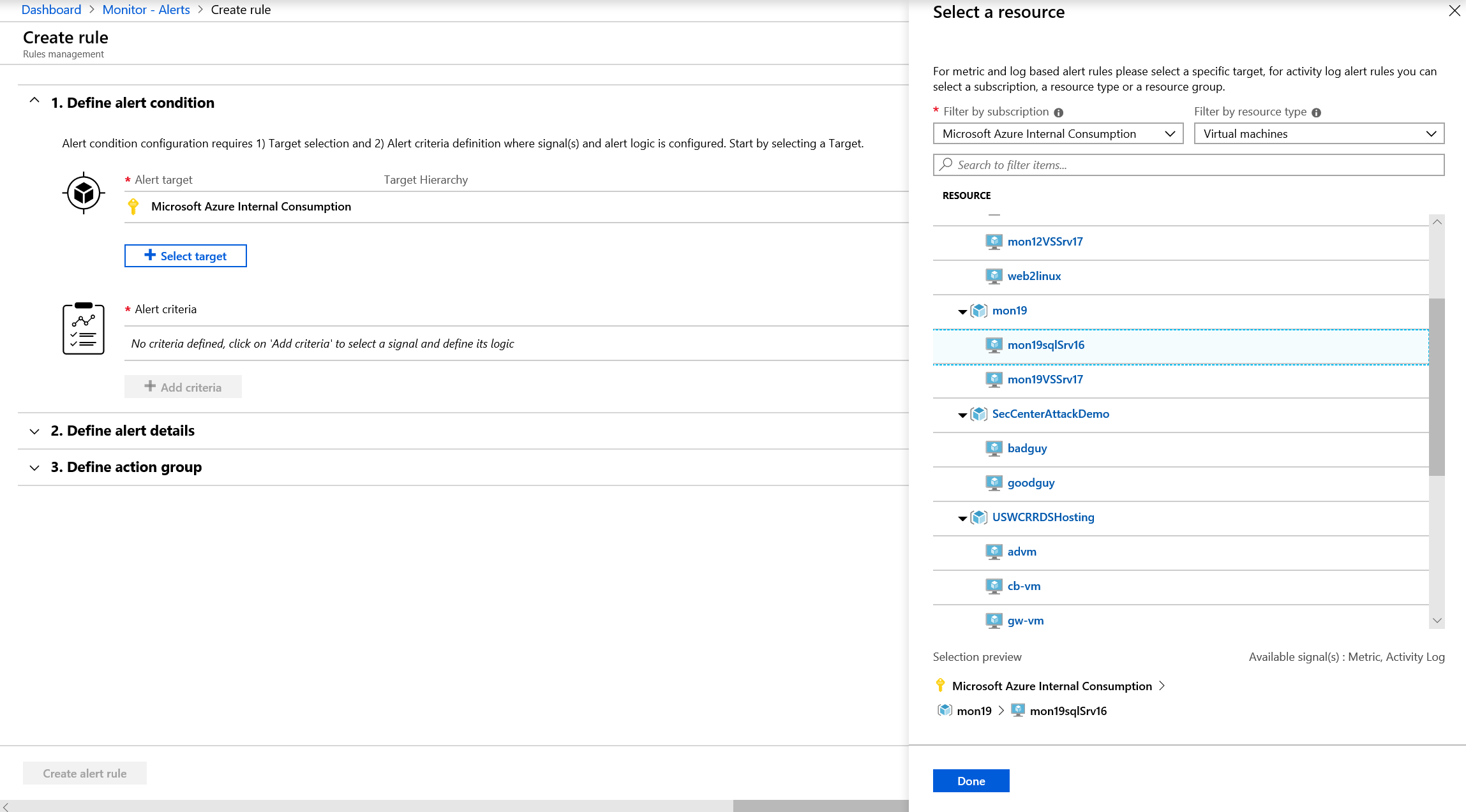
  


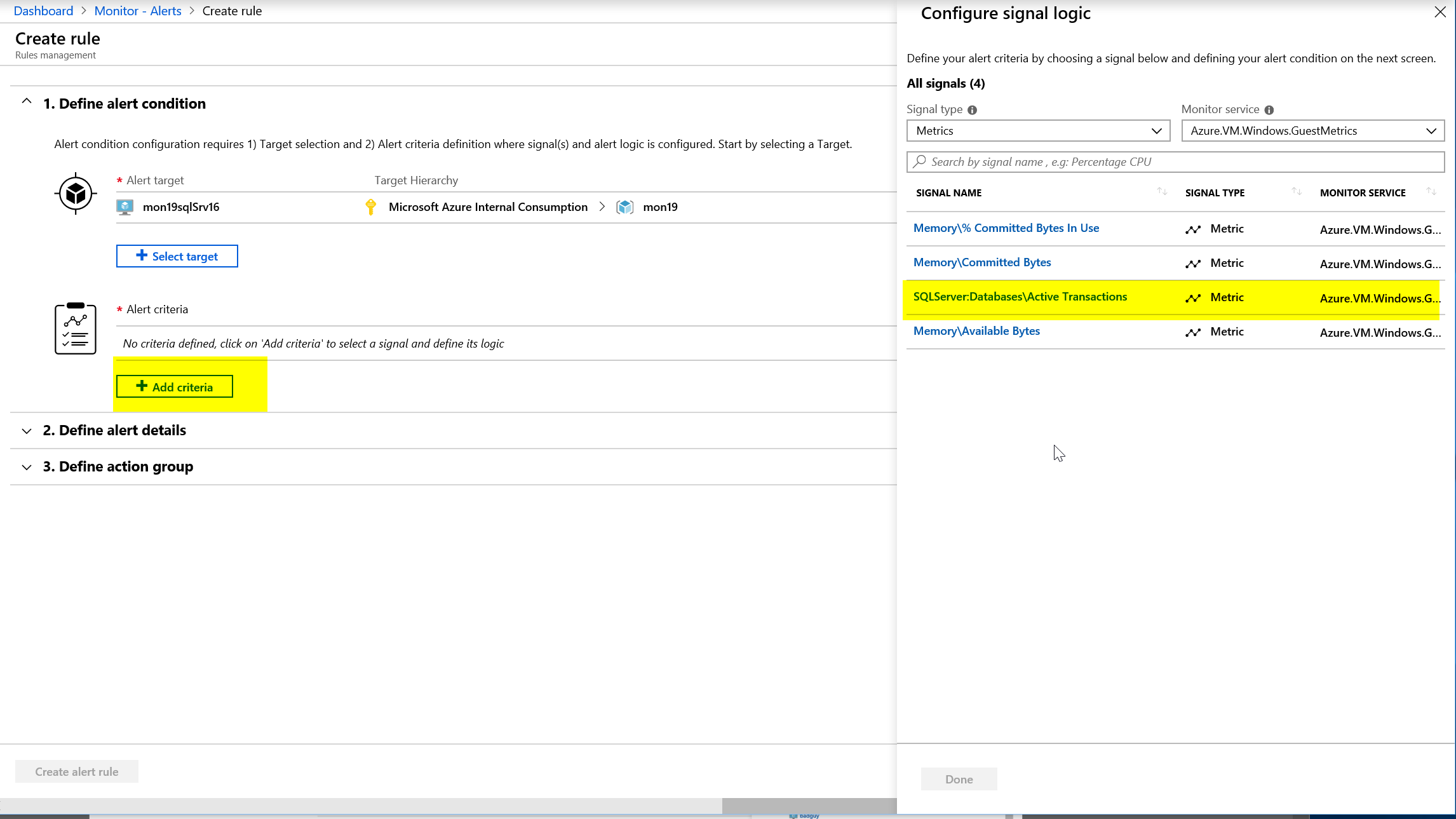
Dashboard should look something like this.

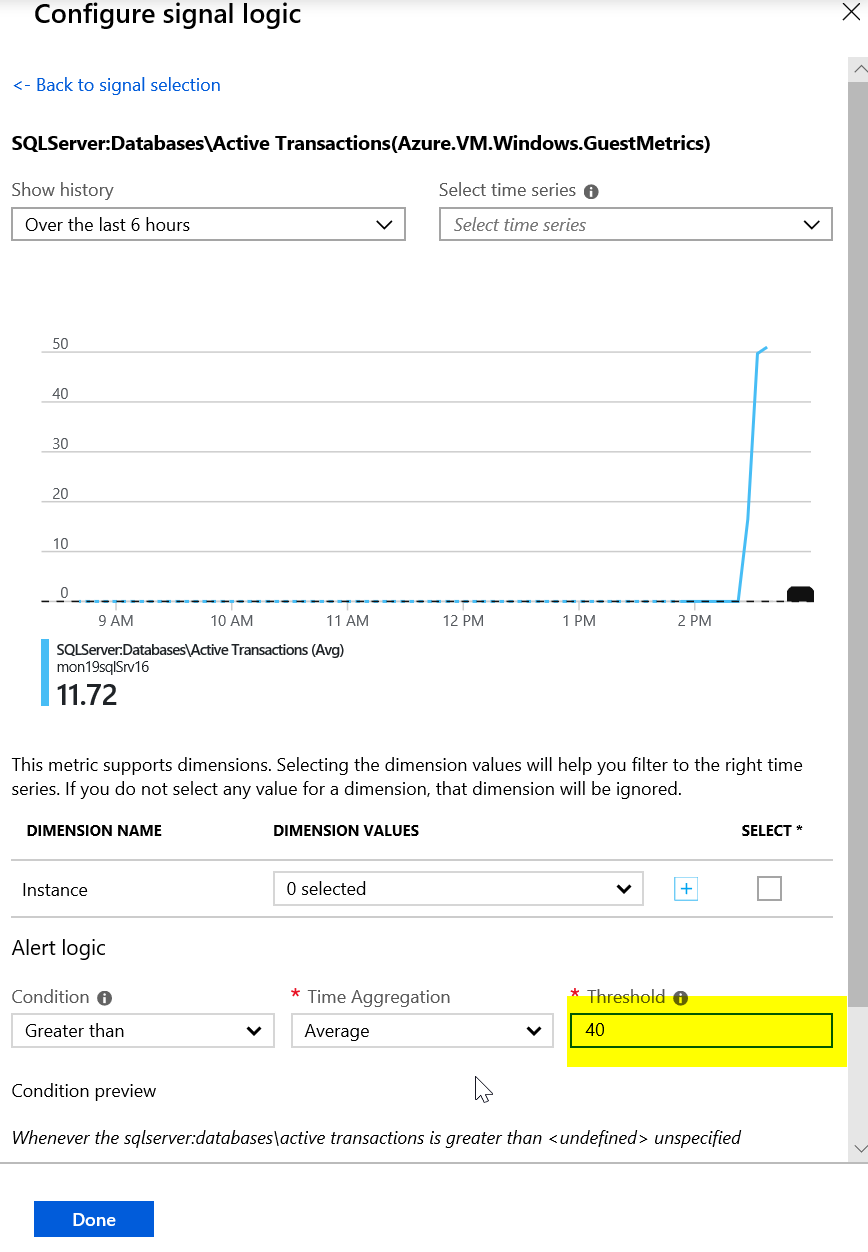
* + From Azure Monitor, create an Action group, to send email to your address

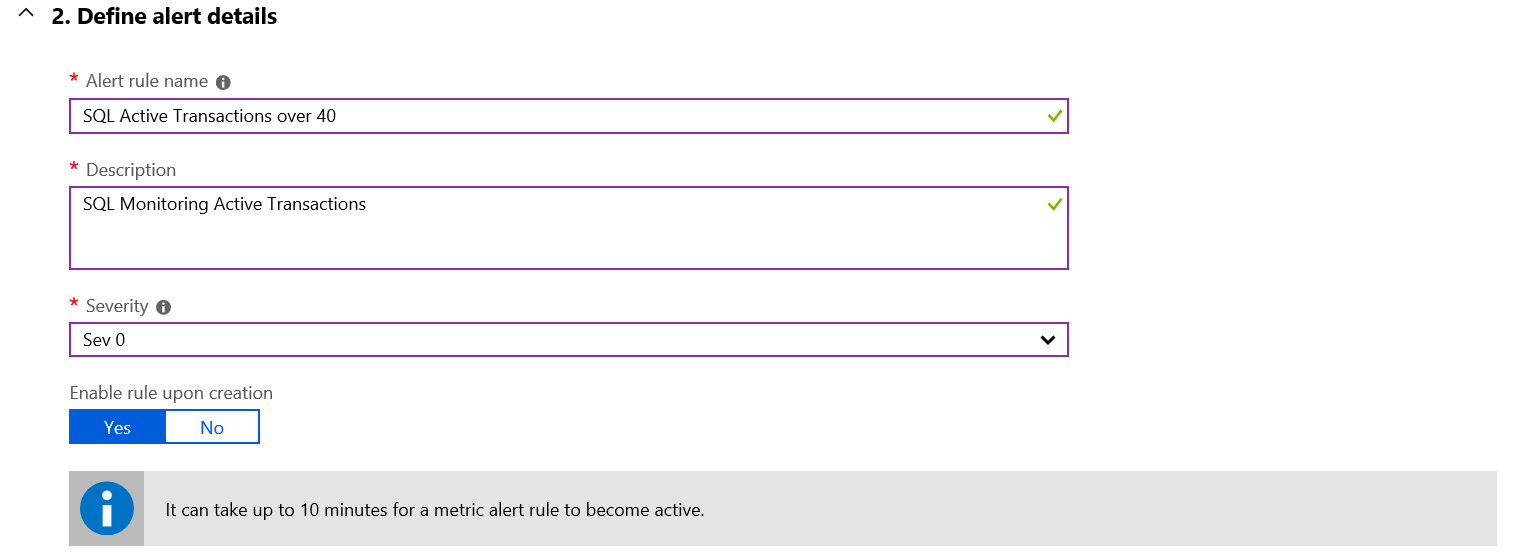


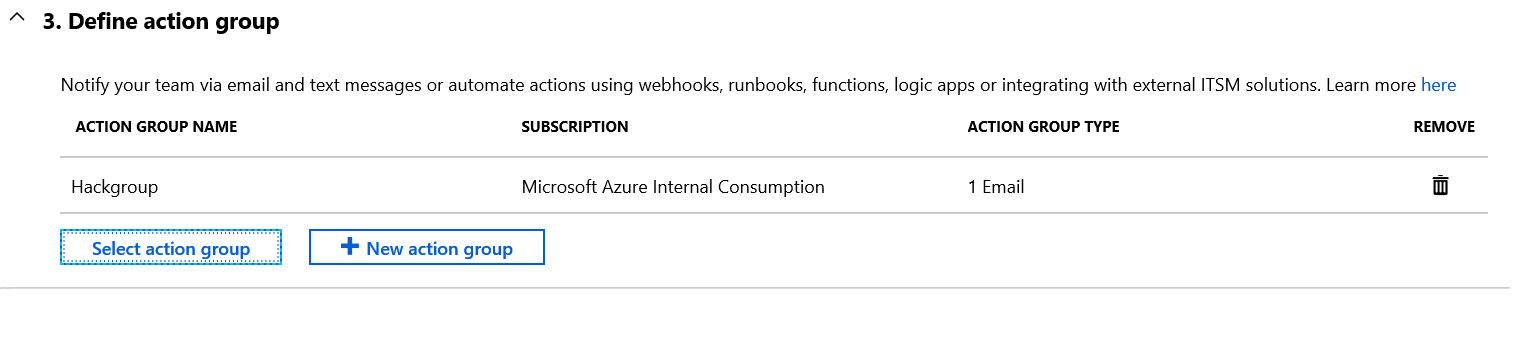
* + Create an Alert if Active Transactions goes over 40 on the SQL Server tpcc database.

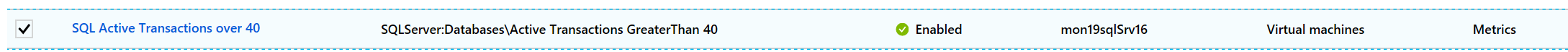


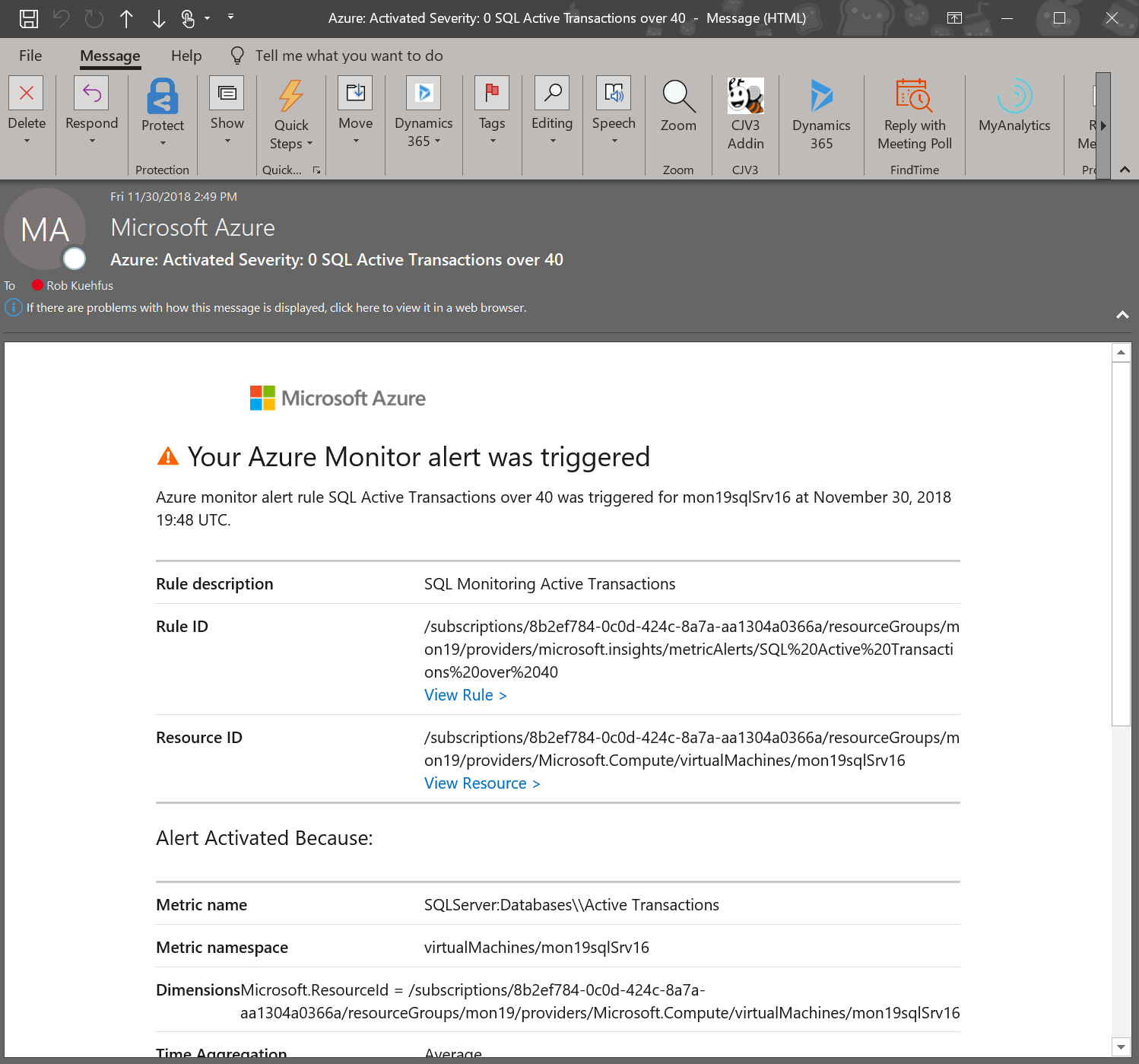






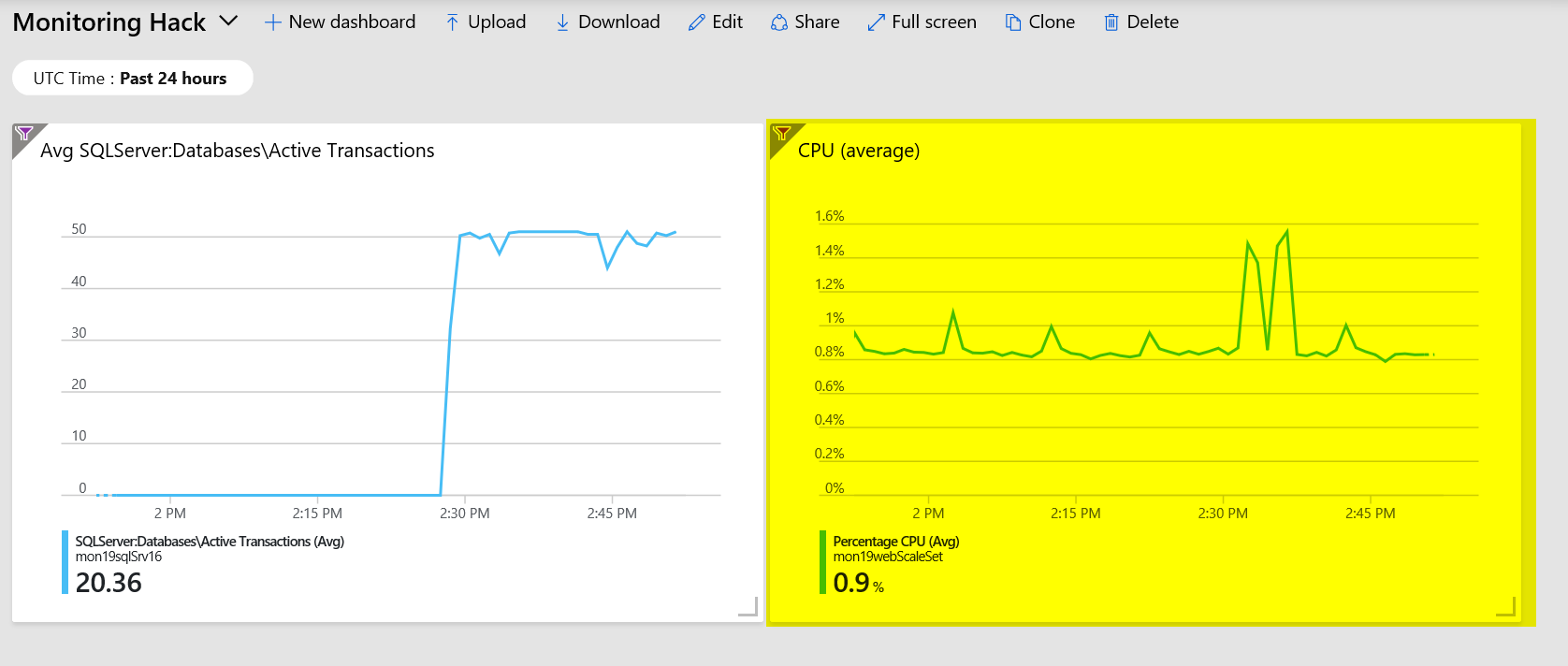






* Create an Alert Rule for CPU over 75% on the Virtual Scale Set that emails me when you go over the threshold.

First create a dashboard to watch the Scale Set CPU



Navigate to the folder called “Loadscripts” under the Resources folder in Student and copy the cpuGenLoadwithPS.ps1 script to both instances running in the Scale Set and run them.

This may be a bit of a challenge to those not used to working with a scale set. If your student just grabs the public IP address and then RDP to it. They will end up on one of the instances but because they are going through the Load Balancer they cannot control which one. Or can they? 😊

If you look at the configuration of the LB it is configured with an inbound NAT rule that will map starting at port 50000 to each instance in the Scale Set. So if they RDP using the PIP:50000 for instance 1 and PIP:50001 for instance 2.

"inboundNatPools": [

{

"name": "natpool",

"properties": {

"frontendIPConfiguration": {

"id": "[concat(variables('webLbId'), '/frontendIPConfigurations/loadBalancerFrontEnd')]"

},

"protocol": "Tcp",

"frontendPortRangeStart": 50000,

"frontendPortRangeEnd": 50119,

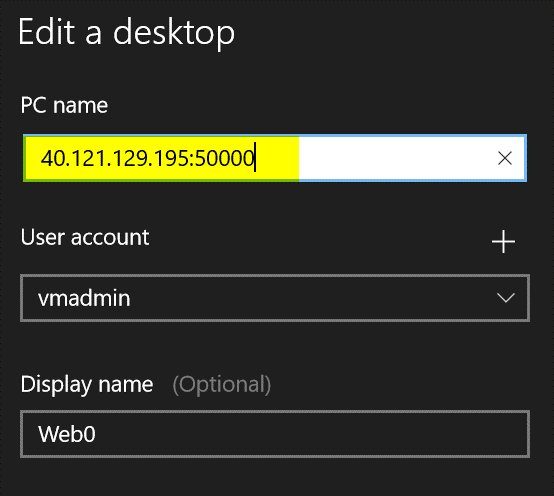
"backendPort": 3389

}

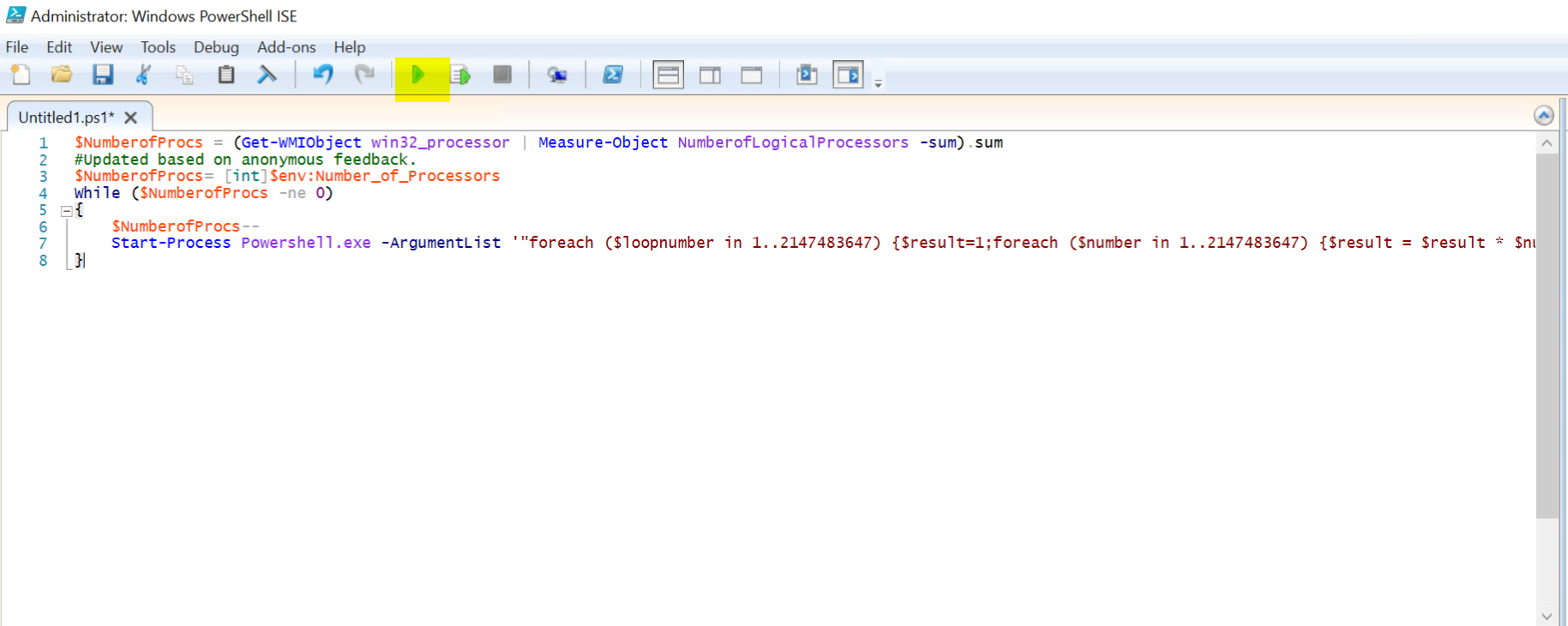
}

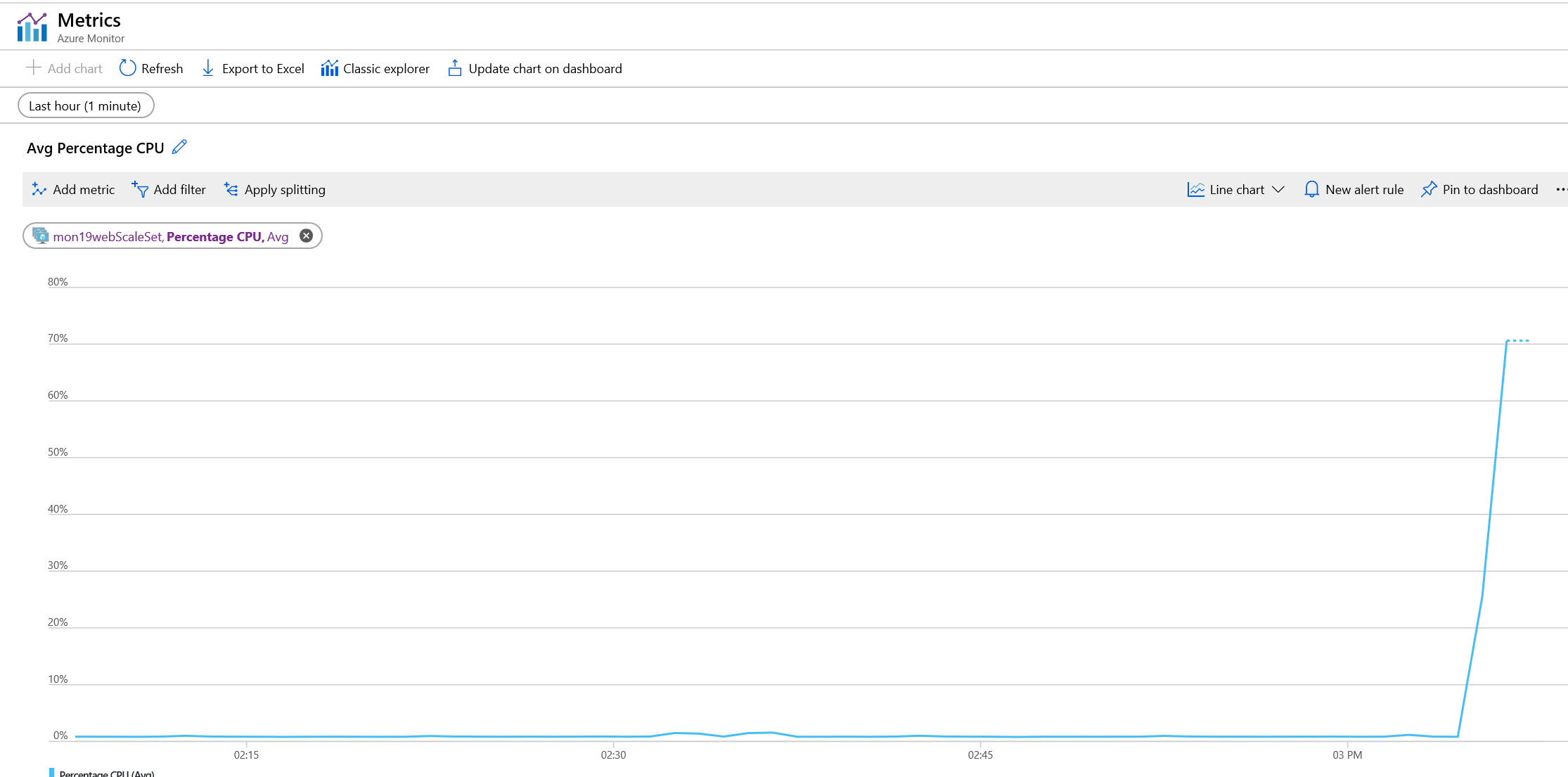
],

For Example,

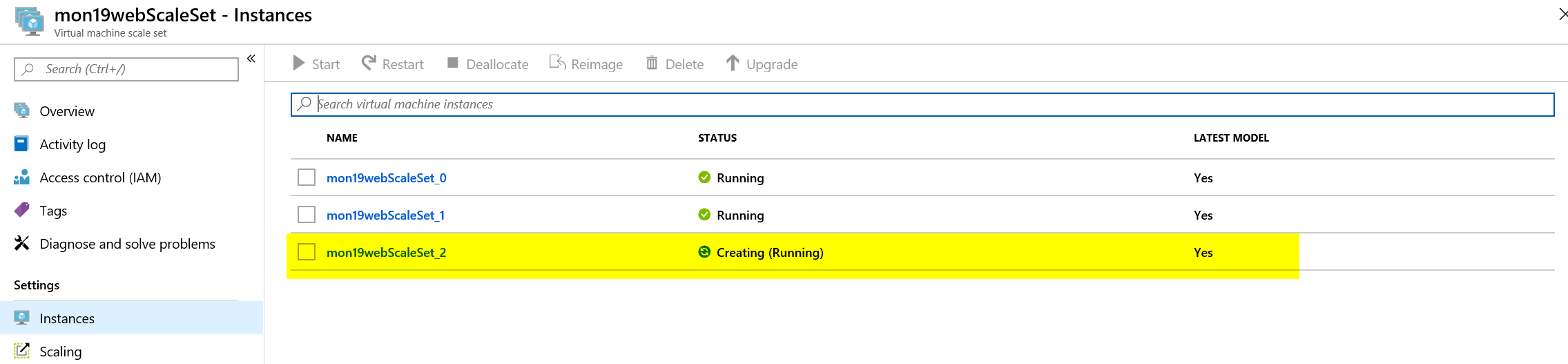


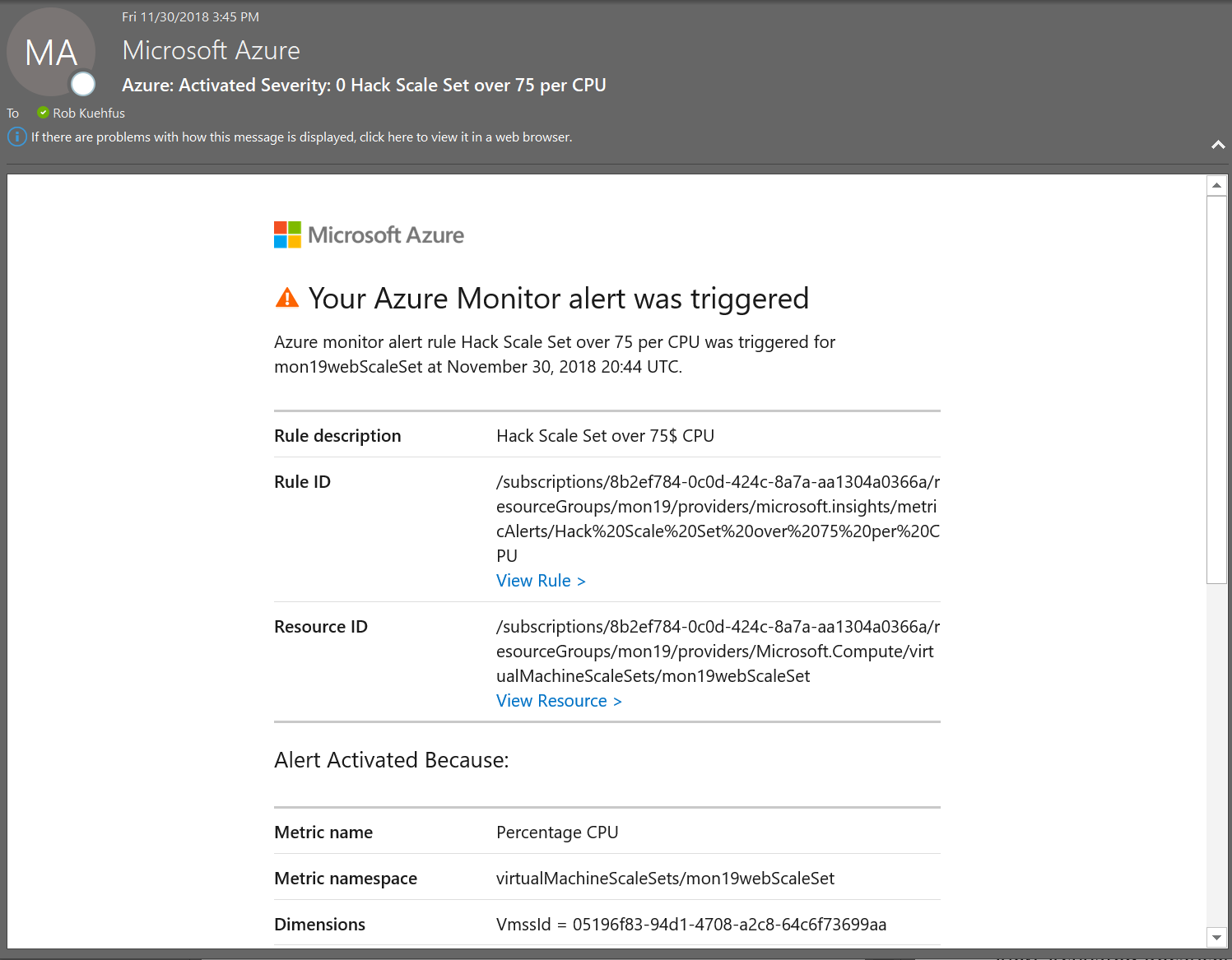
Jump on to both VMs in the Scale Set, Open the PowerShell ISE, Copy the script in the window and run it. You may need to run it more then once to really add the pressure. This script will pin each core on the VM no matter how many you have.





The trick to getting the alert to fire is to pin both instances at the same time as the CPU metric is an aggregate of the scale set. If you just max the CPU on one server to 100% the Scale Set is only at 50% and till not trip the alert threshold of 75%. Also, if they run the script and then setup the Alert Rule then to back to run another test to trip the alert, they have scaled out to a third instance and not realized it. They may need to jump on that box and max it out as well.





* First team to send me both alerts wins the challenge!!
* Good luck!

# Day 2